1. Objection to the Specification

The Title stands objected to in that it is not descriptive of the invention to which the claims are directed. Correction of this defect has been made by the preceding amendment.

2. Objection to the Drawings

The drawings stand objected to as being informal. Formal drawings are therefore submitted herewith under separate Letter to the Official Draftsperson.

3. The §103 Rejection of Claims 1-13, 16-17, 24-25, 27-38, 41-42, 44-47, 49-50, and 52-55

Claims 1-13, 16-17, 24-25, 27-38, 41-42, 44-47, 49-50, and 52-55 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Mukherjee et al. (5,317,729; hereinafter "Mukherjee") in view of Koerber et al. (5,581,755; hereinafter "Koerber").

With respect to claim 1, the examiner asserts that Mukherjee substantially discloses the claimed invention, including the step of "initiating a search request within a sequence of delta streams for a number of data bytes" at col. 12, lines 27-31; the step of "fulfilling the search request with data bytes provided by a first sequence of delta streams" at col. 11, lines 35-38; and the step of "fulfilling the search request provided by a second sequence of data streams" at col. 11, lines 39-43.

Mukherjee discloses a method for storing versioned data which is fundamentally different from the method and apparatus disclosed by applicant. Specifically, Mukherjee teaches a method wherein an entire copy of an object is saved every time the object is versioned. In this manner, "When any attribute of a versioned object is changed, a new instance of the same object is created." Col. 3,

lines 10-12. Thus, Mukherjee does not teach a delta versioning system as taught by applicant. Delta versioning only stores the differences between two discrete versions. Delta versioning does not store numerous complete "instances" of an object, but rather, stores only one complete instance of an object, and numerous delta change files. In order for Mukherjee to retrieve a prior version of an object, all that is required is retrieval of the correct object. With delta versioning, one must assemble an object from one or more deltas. ***Each and every step disclosed in applicant's claim 1 is directed to the assembly of an object. On the other hand, Mukherjee is solely directed to the indexing and retrieval of complete object instances. No assembly is required!***

The examiner asserts that col. 12, lines 27-31 of Mukherjee disclose the first step of applicant's claim 1, namely:

> a) initiating a search request, within the sequential plurality of delta streams, for a number of data bytes to transfer to the updated data stream;

Column 12, lines 27-31 of Mukherjee (as cited by the examiner) are taken from the following claim, which in its entirety reads:

> 10. The method of claim 2 wherein said retrieving step includes:
>
> presenting an interface screen to said authorized user containing user-selectable search query options;
>
> selecting a search query option and identifying a set of input parameters for said search query to collectively define a search strategy;
>
> searching said plurality of versioned data object data files and identified a target plurality of versioned data objects, each versioned data object of said target plurality satisfying said search strategy;
>
> presenting said target plurality of versioned data objects to said authorized user as the result of said search query.

Object assembly steps are entirely absent from this claim. The authorized user merely selects a search strategy identifying a set of input parameters. The query then determines which "object instances" in either a "master item file" or an

"affected item file" satisfy the input parameters (see not only claim 10, but also claims 1 and 2 from which it depends). If an object satisfies the input parameters, it is retrieved *in toto* and presented to the authorized user.

On the other hand, the first step of applicant's claim 1 merely retrieves "a number of data bytes", and not an entire "object". Additional data bytes are then retrieved via additional search requests. Assembly of a file or object is performed as the various data bytes are transferred to an "updated data stream".

The examiner asserts that col. 11, lines 35-43 of Mukherjee disclose the second step of applicant's claim 1, namely:

> b) fulfilling the search request with data bytes provided by the last delta stream in the sequential plurality of delta streams which is capable of supplying data bytes;

Column 11, lines 35-43 of Mukherjee read as follows:

> if said first engineering change satisfies a search query, retrieving said first affected item record and said first modified plurality of versioned data objects by means of said first sequence number; and
> if said second engineering change satisfies a search query, retrieving said second modified plurality of versioned data objects by means of said second sequence number.

Again, each of these steps indicates that whole objects are being retrieved.

Mukherjee discloses a method of versioning engineering changes. An engineering change is defined as a "package" of "designs" or "design modifications", which may include items such as "components in bills of material, optional components, local substitute components", etc. Col. 1, lines 22-55. As a result, a single engineering change may comprise classes and hierarchies of objects, similar to an entire file system of a computer, wherein a single directory of files may have sub-directories, etc. In the above excerpt from Mukherjee's claim 1, Mukherjee is merely stating that if a first engineering change satisfies a search query, all of its components are retrieved at the same time, and if a second engineering change satisfies a search query, all of its components are retrieved at the same time. However, all of the components retrieved are complete objects, and again, no

assembly of objects is required. Although applicant's claims could be modified to claim multiple-object ***assembly***, it is important to note that applicant's claim 1 requires object assembly, and not merely object retrieval.

Applicant agrees with the examiner's assertions that Mukherjee does not disclose 1) the step of searching an original data stream to fulfill a search request if a sequential plurality of delta streams is incapable of fulfilling a search request, and 2) the step of initiating additional search requests until a search request cannot be fulfilled, and an updated data stream is complete. However, applicant does not agree with the examiner's assertion that:

> . . .Koerber disclose[s] an analogous system that searches sequences of data streams stored in a repository, starting with the most recent to the least recent sequences until the search request is found (see col. 13, line 60 - col. 14, line 36). It would have been obvious to one of ordinary skill in the art of data processing to combine the teachings of the cited references because Koerber's system would allow Mukherjee's to more rapidly detect version changes and retrieve a requested data sequence.

Koerber is directed to a method of ***creating*** a versioned database of historical object information, and is not directed to a method of ***retrieving (or reconstructing)*** historical object information. Koerber makes no mention of information retrieval, but for in the context of creating additional versioned objects.

More specifically, Koerber discloses a version service 22d which enables a user "to record ***changes to objects*** over time" while maintaining "information about the previous states" of the objects (Col. 3, line 57 - col. 4, line 1; emphasis added). "In other words, version information provides a history of the system, data and processes for an information system." (Col. 4, lines 1-3). Koerber defines his version service with more particularity in FIG. 2, and column 9, lines 17-46, both of which are reproduced below:

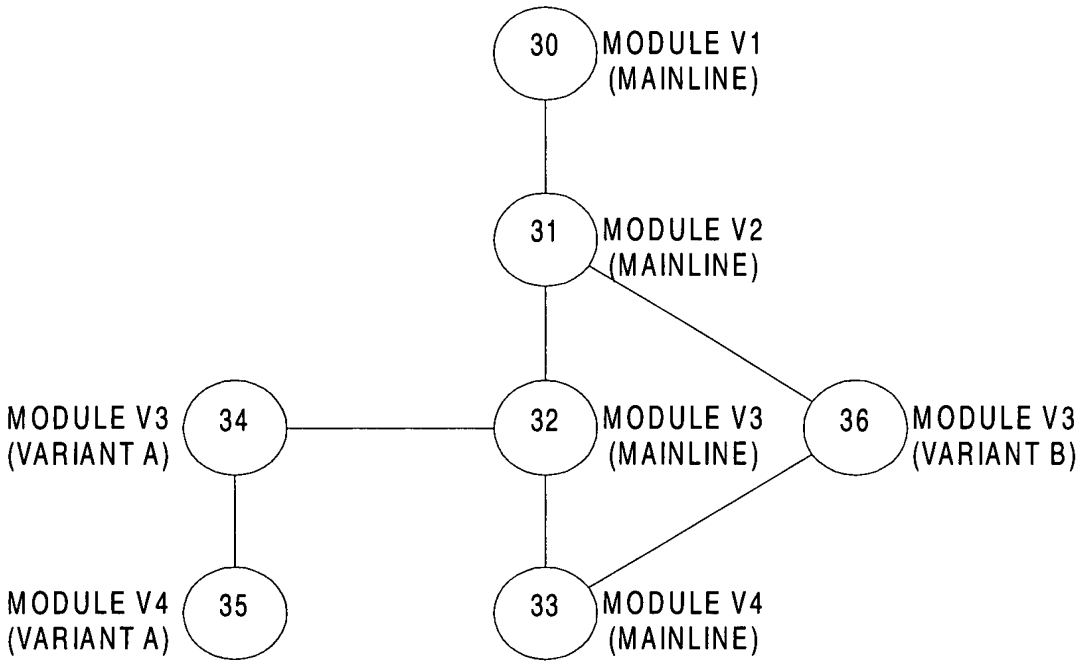

FIG. 2

A Versionable Object is a repository object for which the version service records state changes over time. Each Module in FIG. 2 is a versionable object. ***A Versioned Object*** is a repository object that ***represents a particular historical state of a versionable object. In FIG. 2 each circle stands for a versioned object that represents a particular state of Module.***

A version graph represents the historical states of a versionable object in the repository. FIG. 2 shows an example of a version graph for a versionable repository object called Module. The following definitions will be helpful in understanding this invention when taken in conjunction with the graph shown in FIG. 2.

A Variant Branch, which represents a single line of development, is a subset of versioned objects in a version graph. *The variant branch* is a linear path that *shows the history of changes from one versioned object to another* for a particular line of development. In FIG. 2, the version graph has three variant branches-mainline, variant A and variant B. The mainline branch is made up of objects 30 through 33. The variant A branch is the path including objects 34 and 35 (labeled Module v1 {mainline} and Module v4 {variant}). The variant B branch is the path including object 36 (labeled Module v3 {variant B}).

A Variant is a versioned object on a variant branch. A version graph can contain multiple variants at the same level of

> ancestry. ***The terms Variant and Versioned Object are interchangeable.*** In FIG. 2 there are two variants of Module-objects 34 and 35 on the variant A branch and object 33 on the mainline branch-as the final level (v4) represents in FIG. 2. . . .
>
> Emphasis added.

In summary, FIG. 2 illustrates the structure of Koerber's version service. Each circle in FIG. 2 represents a versioned object, or variant. Each versioned object in turn represents a "particular historical state" of a versionable object. The examiner asserted in a telephone interview held with applicant's representative on July 14, 1997 that each versioned object disclosed in FIG. 2 of Koerber was equivalent to a delta, due to the above language of Koerber which states, "The variant branch. . .shows the history of changes from one versioned object to another. . ." ***When interpreted in context, it is clear that a variant branch does not store any data. Data is only stored within versioned objects, and each versioned object represents a particular historical state of a versionable object.*** That is, each versioned object is a complete object. ***Koerber does not teach or suggest that a versioned object is composed of one or more deltas which are somehow merged upon retrieval of a versioned object.***

For example, if each object is a file, then each versioned object represents a particular historical state of a file.

On the other hand, applicant's disclosure teaches that only a single historical state of a file need be stored in a version service. At various points in time, deltas representing the changes between two successive file versions are stored in the version service. The successive file versions themselves are not stored in the version service. Particular historical states of the file other than the single historical state stored in the version service are then constructed by merging one or more of the deltas with the single preserved historical state.

Other than in the above passage from Koerber, there is no additional discussion of "variant branches". In fact, a variant branch is merely a symbolic representation of the historical progressions between versioned objects. Although the variant branches are shown graphically in the version graph of FIG. 2, there is no

teaching or suggestion in Koerber which indicates that these branches actually exist within version service 22d. Rather, each versioned object is associated with "references", such as indications of whether it belongs to any superclasses (FIG. 4, 56), its version number (FIG. 4, 57), and relationship data such as whether a versioned object is a "parent" of another versioned object (FIG. 10, 143), etc.

Although the examiner argues that Koerber's statement that a "*variant branch. . .shows the history of changes from one versioned object to another*" teaches that variant branches store "something else" not stored within the versioned objects, such as delta information, this is clearly not the case. Although one might use hindsight, having read applicant's disclosure, to modify Koerber such that variant branches do physically exist and store delta information such as that disclosed by applicant, there is clearly no teaching or suggestion within Koerber, nor the art of record as a whole, which would motivate someone to do this. Furthermore, even if such a teaching or suggestion existed, this would only show that a method of ***creating*** versioned repository of delta information existed, which as stated above, is not the same invention as a method of using the delta information to ***reconstruct*** a file as it existed at a given point in time.

In describing his file reconstruction process, applicant frequently mentions the "merging" of delta streams. Koerber also mentions merging, but in an entirely different context. The step of merging objects is illustrated in FIG. 3 of Koerber by process step 46. The process of merging objects is illustrated more fully in FIG. 7, and is described in detail in column 12, lines 31-55. A merge, as defined by Koerber, consists of merging two objects in the same version graph which are not from the same variant branch. For example, the version graph of FIG. 2 shows Module V4 (Mainline) being derived from the merge of Module V3 (Mainline) and Module V3 (Variant B). Koerber clearly states that "Validating the merge request is performed to ***insure. . .that the merge will not cause two objects from the same variant branch to be merged into a new object on the same variant branch."*** Column 12, lines 34-35, and 38-40. Thus, with respect to FIG. 2, Module V3 (Variant A) could not be merged with Module V4 (Variant A). ***This clearly teaches away from the storage of delta information!*** If Koerber had contemplated that

Module V3 (Variant A) and Module V4 (Variant A) could be deltas, then he would not have taught that the two cannot be merged, since the two would have to be merged to fully recreate Variant A at the time Module V4 (Variant A) was created!

As previously discussed, Mukherjee fails to disclose the first two elements of applicant's claim 1. Koerber also fails to teach these elements. Furthermore, Koerber fails to teach the last two elements of applicant's claim 1, namely:

> c) if the sequential plurality of delta streams is incapable of fulfilling the search request with data bytes provided by the original data stream; and
> d) repeating steps a)-c) until a search request cannot be fulfilled, and the updated data stream is complete.

Since Koerber does not disclose delta streams, and does not even utilize delta technology, Koerber cannot be held to teach the steps of "fulfilling a search request with ***data bytes provided by a data stream***", and "initiating repetitive search requests ***within a plurality of delta streams*** for the purpose of ***building an updated data stream***".

Applicant therefore requests that the examiner's rejection of claim 1 be withdrawn.

With respect to claims 2 & 3, the examiner asserts that Mukherjee teaches:

> 1) delta streams comprise a sequence of match data/delta frames (col. 10, lines 53-57);
> 2) the match frames describe matching segments of a delta stream and a prior stream in terms of byte and addresses (col. 10, line 64 - col. 11, line 19);
> 3) the data frames comprise data in a delta stream which does not appear in a prior stream (col. 11, lines 23-34).
>
> 5/22/97 Office Action, p. 5.

Col. 10, lines 53-57 of Mukherjee disclose the step of:

> creating a first ***engineering change notice*** that identifies ***a first plurality of items in a master item file***, said first plurality of items

comprised of at least a first item, said first item affected by a first engineering change;

Applicant is unsure as to which of these elements the examiner is equating with "delta streams comprising match and/or delta data streams". Match frames and delta frames are elements of applicant's versioned data - yet none of the elements cited in the above excerpt from Mukherjee comprise versioned data. Mukherjee states in column 4, lines 33-36, "***The data in engineering change notices file*** 12, ***master item file*** 14 and engineering change affected item file 16 ***are not versioned***. . .". These are the very items recited in the above excerpt from Mukherjee.

Column 10, line 64 - column 11, line 34 of Mukherjee is cited by the examiner as disclosing applicant's match and delta frames. As previously discussed, it is impossible for Mukherjee to disclose such elements. Mukherjee does not disclose ***anything*** related to a delta versioning process.

For the reasons that claim 1 is believed to be allowable, and for the additional reasons cited above, claims 2 & 3 are also believed to be allowable. Applicant therefore requests that the examiner's rejection of claims 2 & 3 be withdrawn.

With respect to claim 4, the examiner asserts that the step of "reading an original data stream directly from a sequential media" is taught by Mukherjee in column 3, lines 12-20. Applicant was unable to find any mention of storage media in this excerpt, and believes that the examiner might have meant to cite column 4, lines 12-20 of Mukherjee. In column 4, Mukherjee refers to "files stored on direct access (non-volatile) storage device (DASD) 40." Kai Hwang and Fayé A. Briggs, in Computer Architecture and Parallel Processing (McGraw Hill 1984), p. 53, state:

> Memories in a hierarchy can be classified on the basis of several attributes. One common attribute is the accessing method, which divides the memories into three basic classes: random-access memory (RAM), sequential-access memory (SAM), and direct-access storage devices (DASDs). . . .In SAMs, information is accessed serially or sequentially. . . DASDs are rotational devices made of magnetic materials where any block of information can be accessed directly.

Thus, a DASD is not equivalent to a sequential media. In fact, applicant does not believe that Mukherjee's system could function in a sequential media environment. Applicant therefore believes that claim 4 is allowable, and that the examiner's rejection of same should be withdrawn.

With respect to claim 5, the examiner asserts that the step of "writing an updated data stream to a sequential media" is disclosed by Mukherjee in column 3, lines 26-30. Once again, applicant cannot find any mention of storage media in this excerpt. The excerpt does refer to a "design sequence number". However, this "sequence number" indicates a design's chronological order with respect to other designs, and provides no indication of what type of media the design is stored on.

As a result, applicant also believes that claim 5 is allowable, and that the examiner's rejection of same should be withdrawn.

With respect to claims 6-13, 16-17, 24-25, 27-38, 41-42, 44-47, 49-50, and 52-55, the examiner indicated that these claims were rejected on grounds similar to those used to reject claims 1-5. Applicant therefore requests that the examiner's rejection of same be withdrawn for the same reasons that the rejection of claims 1-5 should be withdrawn.

4. Other Issues

All claim amendments made herein are made for the purpose of placing claims in an allowable form, and are not made for the purpose of distinguishing the invention over the prior art.

Submitted herewith is a supplemental Information Disclosure Statement, along with a Certification Under 37 CFR 1.97(e).

CONCLUSION

For the foregoing reasons, it is submitted that claims 1-56 are in condition for allowance, and that all outstanding objections to same should be withdrawn.

Respectfully submitted,
KLAAS, LAW, O'MEARA & MALKIN, P.C.

By: _____
Gregory W. Osterloth
Reg. No. 36,232
1999 Broadway, Suite 2225
Denver, CO 80202
Tel: (303) 298-9888
FAX: (303) 297-2266